jamk.fi

Opinnäytetöiden hallintajärjestelmä

Ville Hietakangas

Opinnäytetyö
Toukokuu 2016
Tekniikan ja liikenteen ala
Ohjelmistotekniikan koulutusohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä	Julkaisun laji	Päivämäärä
Hietakangas, Ville	Opinnäytetyö, AMK	7.5.2016
	Sivumäärä 41	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Opinnäytetöiden hallintajärjestelmä

Tutkinto-ohjelma
Ohjelmistotekniikan koulutusohjelma

Työn ohjaaja
Rantala, Ari

Toimeksiantaja
Jyväskylän ammattikorkeakoulu JAMK

Tiivistelmä

Opinnäytetyön tavoitteena oli kehittää Jyväskylän ammattikorkeakoulun IT-instituutin käyttöön sopiva opinnäytetöiden hallintajärjestelmä, joka korvaisi vanhan tavan hoitaa opinnäytetöiden seurantaa. Tällä hetkellä opinnäytetöitä seurataan ja hallinnoidaan verkkolevyllä sijaitsevan Excel-tiedoston kautta. Tästä tavasta aiheutuu useita ongelmia, kuten se, että tiedosto voi olla auki vain yhdellä käyttäjällä kerrallaan. Näin ollen muut eivät pääse käsiksi tiedoston sisältöön.

Opinnäytetöiden hallintajärjestelmän tarkoituksena on ensisijaisesti mahdollistaa usean yhtäaikaisen käyttäjän toimiminen opinnäytetöiden hallinnassa. Myös opiskelijalla on oltava mahdollisuus nähdä oman opinnäytetyönsä tietoja sekä mahdollisuus ladata opinnäytetyöhönsä liittyviä tiedostoja järjestelmään. Opinnäytetyön ohjaajien on tarkoitus päästä ohjaamiensa opinnäytetöiden tietoihin käsiksi ilman, että tarvitsee odotella muita ohjaajia. Lisäksi koulutusohjaajan ja koulutusvastaavien on saatava tietoa opinnäytetöiden tiloista ja arvioiduista valmistumisajoista visuaalisessa muodossa.

Sovellus toteutettiin ASP.NET web-ohjelmistokehyksellä sekä WCF-servicen kautta toimivalla Server-ohjelmalla, joka käsittelee taustalla olevaa MySQL-tietokantaa. Opiskelijat voivat luoda järjestelmässä aihe-ehdotuksia, jotka koulutusvastaava hyväksyy. Tällöin ne muuttuvat järjestelmässä opinnäytetöiksi. Opinnäytetyöohjaajat voivat merkitä etenevään opinnäytetyöhön liittyviä tietoja järjestelmään tallennettuun opinnäytetyö-olioon, ja he voivat myös ladata opiskelijan lähettämiä opinnäytetyöversioita tiedostoina palvelimelta. Myös koulutusohjaaja, sekä koulutusvastaavat voivat luoda kuvaajia tietokantaan tallennettujen opinnäytetöiden valmistumisajoista.

Vaikka Opinnäytetöiden hallintajärjestelmän toteutetun osuuden laajuus täytti opinnäytetyön vaatimukset, jäi järjestelmä silti pahasti kesken. Järjestelmän perustoiminnallisuudet toimivat, mutta paljon toiminnallisuutta jäi myös toteuttamatta. Näitä puutteita korjataan tulevissa opinnäytetöissä.

Avainsanat (asiasanat)
.NET, ASP.NET, Visual Basic, C#, WCF, jQuery

Muut tiedot

jamk.fi

Description

Author	Type of publication	Date
Hietakangas, Ville	Bachelor's thesis	7.5.2016
		Language of publication: Finnish
	Number of pages 41	Permission for web publication: x

Title of publication
Thesis Management System

Degree programme
Software Engineering

Supervisor
Rantala, Ari

Assigned by
JAMK University of Applied Sciences

Abstract

The idea behind Thesis Management System was to create a system for JAMK's IT-institute, which would replace the old way of keeping track of theses. Currently theses are monitored through an Excel file located on a remote hard drive. This causes problems, however, for instance only one thesis supervisor can access the file at once, and by doing so keep all the other thesis supervisors out of the Excel file.

Thesis Management System's purpose is to allow multiple people to be able to access this data simultaneously, as well as to enable students to access their own thesis in the system, and upload files related to their thesis. Thesis supervisors are supposed to be able to see theses they direct without waiting for other thesis supervisors. Additionally both the education coordinator and the education directors should have access to visual information about theses in the system and estimated finish dates.

The application was created with the ASP.NET web application framework, and Server application working over WCF service, linking the whole system to MySQL database. Students can create new subject proposals in the system, and the education coordinator can accept them, upgrading them into theses. Thesis supervisors can keep the information of theses they direct in the system, and also access any files that the students have uploaded on their theses. The education coordinator and education directors can also draw graphs of finishing dates of theses in the system.

Even though the completed part of Thesis Manager system is enough for a thesis, the system is still quite incomplete. Basic functionalities do work, but much of the system was left unfinished. These shortcomings will be fixed by future developers in form of new theses.

Keywords/tags (subjects)
.NET, ASP.NET, Visual Basic, C#, WCF, jQuery

Miscellaneous

Sisältö

Kuviot

Lyhenteet ja termit

3-tier	kolmikerrosarkkitehtuuri. Malli, jossa järjestelmä tai sovellus jakautuu kolmeen osaan: käyttöliittymään, loogiseen ohjelmistoon ja tietokantaan.
Binääritiedostot	Binääritiedostot, lyhyesti binäärit, ovat koodista konekielelle käännettyjä kirjastoja, joita voi käyttää apuna ohjelmaa kehitettäessä.
Client	(Tässä työssä Client-puoli) Järjestelmän web-pohjainen käyttöliittymä, johon kuuluvat käyttäjälle näkyvä liittymä ja sen taustalla toimiva ohjelmakoodi, joka mahdollistaa toiminnallisuuden käyttöliittymällä.
CWS	Client Web Service-rajapinta, jonka avulla Serveriä voi käyttää REST-rajapinnan yli.
Enumeraatiot	Tietokannassa osa tiedoista tallennetaan numerona. Koodia kehitettäessä pelkkä numero ei aina kerro, mitä tietoa tilanteessa halutaan kuvata. Tätä varten enumeraatiot korvaavat numeroarvot ymmärrettävillä sanoilla, jolloin kehitystyö on paljon helpompaa.
Event	Tapahtuma, joka tunnistetaan koodissa. Esimerkiksi hiiren klikkaus on tapahtuma, jonka seurauksena voidaan suorittaa haluttu toimenpide koodissa.

IIS	Internet Information Services, Microsoftin tarjoama ohjelmisto nettisivujen jakamiseen.
IIS Express	IIS:n Visual Studion sisään rakennettu versio, jota voidaan ajaa suoraan kehitysympäristöstä.
Javascript	Dynaaminen komentosarjakieli, jolla voidaan toteuttaa toiminnallisuutta sivuille ilman, että sivua tarvitsee päivittää.
JQuery	Javascriptin päälle rakennettu kirjasto, joka helpottaa Javascriptin käyttöä sivuilla.
Käyttöliittymä	Järjestelmän käyttäjälle näkyvä osuus eli verkkosivut ja logiikka joka tuo tietokannasta tiedot ruudulle oikein.
Labranet	Jyväskylän ammattikorkeakoulun sisäinen verkko tietotekniikan opiskelijoille.
LAMP	Linux-Apache-MySQL-PHP. Linux-käyttöjärjestelmän päälle koottu kokonaisuus, joka sisältää kaiken, mitä yksinkertaisen verkkosovelluksen toteuttamiseen tarvitaan.
MySQL	Relaatiotietokantaohjelmisto, jolla järjestelmän tietokantaa pidetään yllä.

N-tier	Monikerrosarkkitehtuuri. Malli, jossa järjestelmä jakautuu useisiin eri tasoihin. Myös 3-kerrosarkkitehtuuri lukeutuu tähän joukkoon.
Olio	Struktuuri, joka sisältää tietyn kokonaisuuden tiedot.
Palvelin	Sovelluksen takana oleva järjestelmä, joka toimii sovelluksen alustana.
PHP	PHP: Hypertext Preprocessor. Ohjelmointikieli, jolla järjestelmää alettiin alunperin toteuttamaan.
REST	Representational State Transfer, HTTP-protokollan yli toimiva tiedonsiirtotekniikka.
Server	(Tässä työssä Server-puoli) Järjestelmän taustalla toimiva tietokantaan yhteydessä oleva osa.
View	Näkymä tietokannan tauluun tai tauluihin, jolla voidaan rajoittaa, mitä osia taulun kentistä nähdään, tai jolla voidaan lisätä muiden tietokantataulujen kenttiä yhteen tietokantahakuun helposti.
Visual Basic	BASIC-pohjainen ohjelmointikieli.

VPN	Virtual Private Network. Protokolla, jolla esimerkiksi kotona oleva tietokone voidaan yhdistää osaksi jotakin sisäverkkoa.
WCF	Windows Communication Foundation mahdollistaa asynkronisten viestien lähettämisen ja vastaanottamisen Serverin ja käyttöliittymien välillä.
Web Service	Rajapinta, jonka kautta ohjelmalle voi lähettää kutsuja esimerkiksi WCF-kirjaston avulla.

1 Toimeksiantaja ja kehitysympäristö

1.1 Toimeksiantaja JAMK

Jyväskylän ammattikorkeakoulu, JAMK, on vuonna 1994 perustettu kolmannen asteen oppilaitos, joka kouluttaa opiskelijoita monille eri aloille. Pääkampus sijaitsee Jyväskylän keskustan tuntumassa, ja muut koulutusyksiköt ovat ympäri Jyväskylää sekä Saarijärven Tarvaalassa. Oppilaitoksen eri yksiköissä opiskelee yhteensä noin 8500 opiskelijaa, ja valmistuvia on vuosittain noin 1500. Valmistuneista yli 80 % työllistyy seuraavan vuoden aikana valmistumisestaan. (Jyväskylän ammattikorkeakoulun verkkosivut: Tutustu JAMKiin 2016.)

Jyväskylän Ammattikorkeakoulu Oy:n, jonka toimiala on ammattikorkeakoulun ylläpito, omistavat kolme eri osapuolta: Jyväskylän kaupunki, Äänekosken ammatillisen koulutuksen kuntayhtymä POKE ja Jämsän kaupunki. Jyväskylän kaupunki omistaa 90 % yhtiöstä. Kaksi muuta omistajaa jakavat jäljelle jääneen osan puoliksi. (Jyväskylän ammattikorkeakoulun verkkosivut: JAMKin hallinto 2016.)

Oppilaitoksen organisaatio jakautuu viiteen osaan. Nämä ovat ammatillinen opettajakorkeakoulu, liiketoimintayksikkö, hyvinvointiyksikkö, teknologiayksikkö sekä JAMKin hallinto. JAMKin teknologiayksikkö jakautuu kolmelle kampukselle. Nämä ovat pääkampus, Biotalousinstituutti ja Lutakon kampus. Tämän opinnäytetyön tuloksena syntynyt järjestelmä tehtiin tieto- ja viestintätekniikan opiskelijoiden ja opettajien käyttöön. Tieto- ja viestintätekniikka kuuluu teknologiayksikön koulutusohjelmiin. (Jyväskylän ammattikorkeakoulun verkkosivut: Teknologiayksikkö 2016.)

1.2 LabraNet

LabraNet on JAMKin teknologiayksikön opiskelijoiden käyttöön tarkoitettu sisäverkko. Verkkoon on pääsy vain Lutakon kampuksen kerrosten 3. ja 4. koneilta sekä kotikoneilta VPN-yhteyden avulla.

1.3 Toimeksianto ja tavoitteet

Jyväskylän ammattikorkeakoulun IT-instituutti on hoitanut opinnäytetöiden seurannan tähän mennessä ainoastaan verkkolevyllä sijaitsevan Excel-tiedoston avulla. Käyttäjä kirjautuu JAMKin verkkoon, avaa kansion, jossa Excel-tiedosto on, muokkaa tiedostoa ja tallentaa sen. Tämä on hyvin hankala ja jäykkä tapa hoitaa opinnäytetöiden ylläpito. Esimerkiksi jos joku käyttäjistä unohtaa Excel-tiedoston auki omalla etätyöpöydällään, eivät muut voi sitä silloin muokata. Muutenkin prosessi on turhan pitkä ja monimutkainen, varsinkin kun tiedostoon merkitään usein vain yksittäisiä muutoksia. Kun opinnäytetyöohjaajat näkevät kaikki tekeillä olevat työt, on epärelevantin tiedon määrä turhan suuri. Lisäksi opinnäytetyön tekijä ei pääse itse tarkistelemaan työnsä tilaa lainkaan.

Ratkaisuna tähän JAMKin IT-instituutti päätti hankkia verkkoselaimella toimivan version samasta tiedostosta. Tällöin voidaan määrittää, kuka näkee mitäkin, ja yhtä aikaa asioiden tekeminen on mahdollista. Tällöin myös opinnäytetyön tekijä pääsee itse näkemään ainakin suurimman osan opinnäytetyöhönsä liittyvästä datasta.

Varsinaisen järjestelmän toteutus tapahtui kahdessa osassa: itse sovelluksen kehitys sekä palvelinohjelmiston asennus ja konfigurointi sovellusta varten. Minun tehtävänäni oli toteuttaa sovelluksen kehitys, mutta myös palvelimen puolelle piti jonkin verran tutustua, jotta järjestelmä kokonaisuutena toimisi mahdollisimman hyvin.

2 Vaatimukset toteutettavalle sovellukselle

2.1 Yleistä

Järjestelmää voivat käyttää kaikki, joilla on Labranet-tunnukset. Eri käyttäjille annetaan eri rooleja, joiden mukaan käytössä olevat toiminnot määräytyvät. Suurin osa käyttäjistä on opiskelijoita, jotka voivat käyttää vain muutamaa toimintoa. Opinnäytetyön ohjaajat ja muu henkilökunta käyttävät eri osia järjestelmästä, ja yhdellä käyttäjällä voi olla useampi rooli. Käyttöliittymässä on linkkivalikko, johon linkit tulostetaan sen mukaan, mitä rooleja kirjautuneelle käyttäjälle on annettu.

2.2 Profiili-näkymä

Profiili-näkymä on jokaisen käyttäjän käytettävissä, ja linkki siihen on löydyttävä selkeästä paikasta, mutta ei kuitenkaan rooleista luotavien linkkien joukosta. Profiili-sivulla käyttäjä voi muokata perustietojaan, kuten sähköpostiosoitettaan ja puhelinnumeroaan. Näitä tietoja ei haeta eikä tallenneta Labranetin AD-järjestelmään, vaan ne ovat paikallisen tietokannan tietoja. Näin ollen myöskään salasanan vaihtoa ei voi suorittaa tätä kautta, vaan se on tehtävä Labranetin Active Directoryn käyttämillä kontrolleilla, yleensä Windowsin käyttäjähallinnan kautta.

2.3 Opiskelijan näkymä

Opiskelijan päänäkymä jaetaan kahteen osaan. Vasemmalle puolelle tulevat järjestelmään tallennetun opinnäytetyön tiedot, joista opiskelija voi muokata joitain perustietoja. Oikealle puolelle tulee lista opinnäytetyöhön liittyvistä tiedostoista sekä mahdollisuus uusien tiedostojen lataamiseen. Nämä tiedostot ovat opinnäytetyön kirjoitusosuuden versioita, joita opiskelija voi ladata liitteeksi opinnäytetyölle.

Opiskelijan näkymän vasemmalla puolella on aluksi vain linkki uuden opinnäytetyön aihe-ehdotuksen lisäämiseksi järjestelmään, sillä uudella käyttäjällä ei vielä ole järjestelmässä olemassa olevaa opinnäytetyötä. Poikkeuksena tästä on se, jos joku henkilökunnan jäsen on syystä tai toisesta luonut opinnäytetyön opiskelijan puolesta. Kun opinnäytetyön aihe-ehdotus on tallennettu tietokantaan, uuden työn luominen ei saa enää olla mahdollista. Luotu aihe-ehdotus näytetään opiskelijalle "Uusi aihe-ehdotus"-linkin sijaan. Opinnäytetöitä ei voi poistaa järjestelmästä luomisen jälkeen, lukuun ottamatta admin-roolin omaavaa käyttäjää.

Toiselle välilehdelle tulee lista töistä, joiden opponoijaksi kirjautunut käyttäjä on määrätty. Tähän listaan tulee todennäköisesti vain yksi rivi, koska yhdelle opiskelijalle yleensä määrätään vain yksi työ opponoitavaksi. On kuitenkin mahdollista, että joissain tapauksissa opiskelijalla voi olla useampikin työ opponoitavana. Klikatessaan listalla olevaa opponoitavaa työtä käyttäjälle avataan työn perustiedot uudessa ikkunassa, jonka kautta voi myös ladata tiedostoja palvelimelle. Nämä tiedostot ovat varsinaisia opponointeja joko .doc-, .docx- tai .pdf -formaatissa.

2.4 Opinnäytetyön ohjaajan näkymä

Ohjaajan näkymässä näytetään työt, joihin hänet on merkitty joko ensimmäiseksi tai toiseksi ohjaajaksi. Ohjaajan listalla saavat näkyä vain ne työt, joiden aihe-ehdotuksen on koulutusvastaava hyväksynyt. Ohjaaja pystyy muokkaamaan opinnäytetyön etenemiseen liittyviä tietoja, kuten esimerkiksi määrittämään arvosanaehdotuksen, valitsemaan työlle opponoijan sekä syöttämään opinnäytetyön valmistumispäivämäärän.

Opinnäytetyöt näytetään ohjaajan sivulla listana, jonka rivejä klikkaamalla voi avata opinnäytetöitä muokkausikkunaan. Muokkausnäkymästä on myös voitava ladata opinnäytetyöhön liittyviä tiedostoja. Nämä tiedostot ovat varsinaisen opinnäytteen kirjoitustyön versioita joko .doc-, .docx- tai .pdf -formaatissa.

2.5 Koulutusvastaavan näkymä

Ensimmäisellä välilehdellä koulutusvastaavan on voitava nähdä kaikki oman koulutusalansa opinnäytetöiden aihe-ehdotukset, hyväksytyt opinnäytetyöt sekä valmistuneet työt. Aihe-ehdotusten tai opinnäytetöiden perustietojen, kuten työn nimen, asiakkaan tai tekijän, muuttaminen ei tule olla mahdollista koulutusvastaavalle. Opinnäytetyöt ja aihe-ehdotukset näytetään koulutusvastaavalle esimerkiksi listana. Listan riviä klikkaamalla aukeaa ikkuna, jossa näytetään valitun työn tiedot. Tästä ikkunasta koulutusvastaava voi määrätä työlle 1. ja 2. ohjaajan sekä hyväksyä aihe-ehdotuksen. Aihe-ehdotuksen hyväksyminen tarkoittaa käytännössä yhden totuusarvon vaihtamista työlle. Jos totuusarvo on false, käsitellään tietokantaan tallennettua riviä aihe-ehdotuksena, ja jos arvo on true, käsitellään sitä opinnäytetyönä.

Koulutusvastaavan on myös saatava näkyville oman koulutusalansa opinnäytetöistä piirretty kuvaaja. Kuvaajasta koulutusvastaava näkee töiden tilanteen kuusi kuukautta eteenpäin. Menneisyydestä näytetään valmistuneiden töiden valmistumispäivämäärät ja tulevaisuudesta töiden arvioidut valmistumispäivämäärät. Näin koulutusvastaava saa visuaalisen näkymän opinnäytetöiden valmistumistahdin trendiin.

2.6 Koulutuspäällikön näkymä

Koulutuspäällikön on nähtävä laajempi kuva opinnäytetöiden sekä opiskelijoiden valmistumisen tilasta. Koulutusvastaavasta poiketen koulutuspäällikön on saatava näkyvilleen kaikkien koulutusalojen opinnäytetöistä ja opiskelijoiden valmistumisajoista kerättyä dataa ja kuvaajia.

2.7 Opintosihteerin näkymä

Opintosihteerin tehtäviin kuuluu opinnäytetyön ohjaajien ohjausmäärien ylläpito ja tapaamisajankohtien syöttäminen järjestelmään. Koulutusvastaava käyttää opinnäytetyön ohjaajien ohjausmääriä määrätessään ohjaajille opinnäytetöitä ohjattavaksi.

2.8 Admin-näkymä

Ylläpitäjän näkymässä on voitava muokata kaikkia tärkeimpiä tietoja sekä opinnäytetöiden että rekisteröityneiden käyttäjien osalta. Ainoastaan ylläpitäjällä on mahdollisuus poistaa opinnäytetöitä, mutta se ei kuitenkaan ole tämän näkymän tarkoitus. Oletusarvoisesti kaikki luodut opinnäytetyöt säilytetään järjestelmässä ikuisesti tai ainakin mahdollisimman pitkään tilastojen keräämistä varten.

Käyttäjien tiedot, kuten tallennetut sähköpostiosoitteet, puhelinnumerot ja opinnäytetöihin liittyvät päivämäärät, ovat ylläpitäjän muokattavissa. Myöskään tähän ylläpitäjän rooli ei ole ensisijaisesti tarkoitettu, mutta ylläpitäjällä on oltava oikeudet kaikkien perustietojen muokkaamiseen ilman tarvetta tietokannan suoraan muokkaamiseen.

Tärkein ylläpitäjän tehtävä on käyttäjien roolien määrittäminen ja tietokannan tietojen oikeellisuuden valvominen. Kovin aktiivisesti ylläpitäjän ei tarvitse oikeuksiaan käyttää, sillä jokainen uusi käyttäjä saa oletuksena opiskelijan roolin. Koska uusia opettajia tai muita henkilökunnan jäseniä erikoisrooleilla tulee vain harvoin, ei aktiivista ylläpitoa tarvita.

3 Teknologiavalinnat

3.1 Yleistä

Kehitystyön aikana käytettiin kahta pääasiallista teknologiamallia. Ensimmäisenä kehitettiin LAMP-järjestelmän päällä toimivaa PHP/MySQL web-sivustoa, joka vaihdettiin ASP.NET/Web Forms -tekniikalla toteutettuun laajempaan järjestelmämalliin. Teknologiat, joita ei tarvinnut vaihtaa tekniikan vaihdon myötä, olivat Javascript/JQuery ja MySQL-tietokanta.

3.2 Ensimmäisen version teknologiat

Järjestelmästä tehtiin käytännössä kaksi erillistä versiota. Ensimmäinen versio toimi 3-tier teknologialla, joka rajoitti sen laajennettavuutta. Lisäksi järjestelmä oli jokseenkin sekava ja hankalasti ylläpidettävä. Rajapinnan teko ulkoisille järjestelmille olisi ollut mahdollista myös tähän malliin, mutta 3-tier mallin ja PHP-kielen hylkääminen ja siirtyminen paremmin tuetun teknologiakokonaisuuden piiriin paransi ohjelmiston tulevaisuudennäkymiä.

PHP

PHP on verkkosivuja kehittäessä hyvin käytännöllinen ohjelmointikieli. Se on kevyt, ja erittäin helppo sisällyttää verkkosivuille. Verrattuna .NET- tai Java-ympäristöihin se on kuitenkin liian suppea monimutkaisempien ja laajojen toiminnallisuuksien tehokkaaseen kehittämiseen. Tästä syystä PHP sopii loistavasti yksinkertaisten 2- ja 3-tier ohjelmien kehittämiseen, mutta huonommin useamman tason ohjelmointiin.

3.3 .NET

3.3.1 Yleistä

Microsoftin kehittämä ohjelmistokomponenttikirjasto .NET otettiin käyttöön PHP-teknologian sijaan ensisijaisesti sen suomien laajennusmahdollisuuksien vuoksi. Lisäksi .NET sopi paremmin laajemman järjestelmän kehittämiseen, sillä .NET-teknologian mukanaan tuoma syntaksi ja toiminnot olivat suurelta osin entuudestaan tuttuja. Toisena potentiaalisena vaihtoehtona kehitysteknologiaksi oli Java EE, jolla

voi myös toteuttaa verkkosivupohjaisia järjestelmiä. Tämä tekniikka olisi toiminut myös paremmin jo olemassa olevan LAMP-serverin päällä.

Valinta .NET- ja Java EE -tekniikoiden välillä oli kuitenkin melko helppo, sillä oma tietotaitoni .NET-tekniikan osalta on hyvin paljon kattavampi kuin Java EE-tekniikassa. Vaikka Java EE sopisi luontaisesti paremmin Linux-ympäristöön, toteutusteknologian valitseminen olemassa olevan serverin perusteella on mielestäni hieman väärä tapa ratkaista ongelma. Varsinkin kun sekä Windows- että IIS-palvelimien pystyttäminen on erittäin helppoa. Lisäksi olisi ollut turhaa vaikeuttamista valita kahdesta tekniikasta se, jonka tuntemus toteuttajalla on lähes olematon.

3.3.2 Visual Basic

Visual Basic on järjestelmän kehityksessä suurimmaksi osaksi käytetty kieli. Serveri, SharedCodes ja CWS ovat kaikki toteutettu Visual Basicilla. Visual Basic on viime aikoina tullut minulle parhaiten tutuksi, ja siksi se valittiin järjestelmän pääasialliseksi toteutuskieleksi. Mitään suurta toiminnallista tai teknistä eroa C#-kielellä ja Visual Basicilla ei ole, lukuun ottamatta muutamia pieniä eroavaisuuksia.

Visual Basicin syntaksi eroaa muista ohjelmointikielistä suuresti. Muissa kielissä sulut ja aaltosulut sekä puolipisteet ovat hyvin suuressa roolissa. Visual Basic taas koostuu lähes selväkielisistä lauseista. Esimerkiksi yksinkertainen ehtolause voi näyttää seuraavalta:

```
Dim b_muuttuja As Double = GetMuuttujanBooleanArvo()
Dim s_muuttuja As String = GetMuuttujanStringArvo()
Dim tulos;

If Not muuttuja AndAlso s_muuttuja = "Value" Then
    tulos = "Value annettu ja boolean oli false"
Else If String.IsNullOrEmpty(s_muuttuja) Then
    tulos = "String on tyhjä."
End If
```

Lisäksi olion tietojen mappaus, eli tietojen asettaminen tai hakeminen massana, on Visual Basicissa kätevämpää kuin C#-kielessä With-struktuurin avulla. With-struktuuria voi käyttää olion täyttämiseen esimerkiksi seuraavalla tavalla:

```
Dim olio As OlioLuokka = GetOlioJostain()
```

```
Dim toinenOlio As New ToinenOlioLuokka() With {.TestiProperty =
"Testidata", .ToinenTestiProperty = 200}

With olio
    .JokinProperty = "Arvo"
    .ToinenProperty = 13
    .BooleanProperty = False
    .ToinenOlio = toinenOlio
End With
```

Vaikka SharedCodes toteutettiin myös Visual Basicilla, sen tarjoamat luokat ja enumeraatiot toimivat myös Client-puolen C#-koodissa, koska niitä ei käytetä suoraan, vaan käännettyinä kirjastoina.

3.3.3 C#

Client-puoli toteutettiin C#-kielellä pääasiassa kahdesta syystä. Tärkeämpi näistä oli se, että valtaosa internetin kautta löytyvistä ASP.NET-sivujen kehittämiseen liittyvistä ohjeista oli juurikin C#-kielellä kirjoitettuja. Toisena syynä oli opinnäytetyön sisällön laajentaminen, ja oman oppimisen lisääminen myös C#-pohjaisen verkko-ohjelmoinnin osalta.

Kaikkien käyttöliittymällä näkyvien sivujen toiminnallisuudet, ja muutamat muut taustalla tapahtuvat toiminnot, on toteutettu C#-kielellä. Esimerkkinä tästä opinnäytetyön ohjaajan näkymässä olevan opinnäytetyölistan täyttäminen hakukriteerien perusteella haetulla datalla, kun hakupainiketta painetaan:

```
protected void searchFirstDirectorThesesButton_Click(object sender,
EventArgs e)
{
    ThesesCriteria crit = new ThesesCriteria();
    crit.FirstDirector = ((UserModel)Session["logged"]).UserId;
    crit.Approved = true;
    if
(!string.IsNullOrEmpty(FirstDirectorSearchThesisStartDayStart.Text))
{ crit.StartDateStart =
DateTime.Parse(FirstDirectorSearchThesisStartDayStart.Text); }
    if
(!string.IsNullOrEmpty(FirstDirectorSearchThesisStartDayEnd.Text)) {
crit.StartDateEnd =
DateTime.Parse(FirstDirectorSearchThesisStartDayEnd.Text); }
    if
(!string.IsNullOrEmpty(FirstDirectorSearchThesisSecondDirector.Select
edValue)) { crit.SecondDirector =
FirstDirectorSearchThesisSecondDirector.SelectedValue; }

    FirstDirectorThesesModel.Load(crit);
    Session["FirstDirector_ThesesCriteria"] = crit;

    FirstDirectorThesesModelGrid.SetValues(FirstDirectorThesesModel);
```

```
    Session["FirstDirector_Theses"] = FirstDirectorThesesModel;

    FirstDirectorThesesGrid.DataSource =
FirstDirectorThesesModelGrid.thesisList;
    FirstDirectorThesesGrid.DataBind();
}
```

3.3.4 Mod_mono

Alkuperäisen kehitystyön myötä Labranetin web-palvelimelle oli jo asennettu LAMP, mutta .NET frameworkkiin siirtyminen vaati alleen palvelimen, joka osaisi ajaa myös ASP.NET sovelluksia ja sisältäisi tuen Web Serviceille. Näin ollen piti keksiä ratkaisu uusien teknologioiden tukemiseen. Vaihtoehtona olisi ollut Windows Serverin pystyttäminen, mutta uuden ympäristön pystytykseen olisi kulunut aikaa ja vaivaa. Pienen pohdinnan ja neuvonpidon jälkeen päätettiin jo olemassa olevaa ympäristöä laajentaa Mod_mono-nimisellä Apachen laajennuksella.

Mod_mono on suhteellisen suppea lisäosa, joten se ei tarjoa kaikkia samoja mahdollisuuksia kuin Microsoftin IIS. Se on kuitenkin riittävän kattava opinnäytetyöjärjestelmän ylläpitämiseen. Tärkeimmät tarvittavat toiminnot ovat tietenkin .NET frameworkin tuki, WCF-teknologian tuki sekä mahdollisuus REST-viestien käsittelyyn. (Mod_mono compabilities 2016.)

3.4 Javascript

Javascript mahdollistaa dynaamisten toimintojen lisäämisen HTML-pohjaisille verkkosivuille. Järjestelmässä on muutamia osia, jotka vaativat Javascriptiä, jotta sivut toimisivat kitkattomasti. Esimerkiksi kaikki dialogit web-käyttöliittymällä toteutetaan Javascriptin avulla. Nappia painettaessa viesti lähetetään Clientillä olevalle sisäiselle WCF-servicelle, joka käsittelee viestin sisällön ja lopuksi sivu päivittyy Javascriptin saadessa vastausviestin WCF-serviceltä.

JQuery

JQuery on laajennus perus Javascript-kirjastoon ja se tarjoaa valmiita funktioita monimutkaisempien toimintojen suorittamiseen. Myös HTML-elementtien käsittely onnistuu JQueryn avulla helpommin kuin pelkässä Javascriptissä.

Lisäksi verkkosivuilla käytettävä JQuery UI -niminen laajennus käyttää JQueryn komentoja. JQuery UI käyttää JQueryn funktioita vielä laajempien kokonaisuuksien

toteuttamiseen, kuten kokonaisten dialogien ja web-komponenttien toiminnallisuuksien rakentamiseen ja tapahtumankäsittelyyn.

Esimerkki JQueryn tarjoamasta yksinkertaistetusta elementtien käsittelystä:

```
//HTML-koodi
<div id="laatikko"></div>

//Javascript
document.getElementById("laatikko").innerHTML = "Test";

//JQuery
$("#laatikko").html("Test");
```

3.5 MySQL

Tietokantateknologioista MySQL on yksi kevyimpiä. MySQL oli luonteva valinta ohjelmiston tietokanta-arkkitehtuuriksi LAMP-järjestelmän takia. PHP ja MySQL toimivat suoraan yhdessä, mutta .NET-arkkitehtuuri ei itsessään tue MySQL-tietokantoja. .NET-ympäristöihin tarjolla olevalla MySQL-kirjastolla tietokannan käyttö kuitenkin onnistuu yhtä helposti kuin vaikkapa MSSQL tai Oracle -tietokantojen käyttö.

Tietokannan hallintaa varten luotiin oma manageri, jolle voi lähettää kyselyn ja listan parametreja. Takaisin palautuu tietyssä muodossa olevaa dataa, joka puretaan tietokantamanageria kutsuneen luokan sisällä ja data mapataan olioihin.

3.6 LDAP

Yksi järjestelmän alkuperäisistä määrittelyistä sisälsi vaatimuksen Labranetin tunnusten tukemisesta. Koska ohjelmiston käyttäjien salasanoja ei tallenneta järjestelmään, tarkistetaan sisään kirjautuvan käyttäjän tunnukset sisään kirjautumisen yhteydessä LDAP-tarkistuksella. LDAP-tarkistus sisään kirjautuessa ottaa yhteyden Labranetin Active Directory -palvelimeen ja tarkistaa annetun käyttäjätunnuksen ja salasanan oikeellisuuden. Tieto onnistuneesta, tai epäonnistuneesta käyttäjätunnuksen ja salasanan tarkistuksesta palautetaan kutsuneelle koodille. Mitään muuta tietoa Active Directory -tilistä ei palauteta järjestelmään, kuten henkilökohtaisia tietoja. Jos käyttäjätunnus ja salasana ovat

oikein, luodaan käyttäjälle kirjautumissessio järjestelmään, jonka olemassaolon ajan käyttäjä voi käyttää järjestelmää.

Labranet-verkon ylläpitäjät tarjosivat ensimmäisen järjestelmäversion kehityksen aikana ohjeet, sekä puutteellisen pätkän PHP-koodia, joiden avulla LDAP-yhteys piti saada toimimaan. Pienen tutkiskelun jälkeen oikeanlainen koodinpätkä saatiin aikaan ja toimintaan. Teknologiamuutoksen myötä yhteys piti kääntää toimimaan .NET-ympäristöstä. Asia ratkaistiin seuraavalla koodilla:

```
Private Function LdapCheck(ByVal userName As String, ByVal userPass
As String) As Boolean
    Try
        Dim de As New DirectoryEntry("LDAP://labranet.jamk.fi",
userName, userPass, AuthenticationTypes.Secure)
        Dim ds As DirectorySearcher = New DirectorySearcher(de)
        ds.FindOne()
        Return True
    Catch ex As Exception
        Throw New LdapConnectionException("Incorrect password.", ex)
    End Try
End Function
```

4 Sovelluksen toteutus

4.1 Toteutusprosessi

Järjestelmää kehitettiin aluksi kotikoneella, mutta pian se siirrettiin JAMKin Labranettiin perustetulle virtuaalikoneelle. Tällöin sitä päästiin kehittämään suoraan lopullisessa ympäristössään. Labranetin virtuaalikoneelle oli valmiiksi asennettu Linux ja LAMP ja SFTP-yhteydet oli konfiguroitu jo valmiiksi toimiviksi, joten itse järjestelmän kehittämisessä päästiin tosi nopeasti alkuun.

Serverille ei ollut pääsyä muualta kuin Labranet-verkon sisäpuolelta. Tämä ei varsinaisesti haitannut, mutta kotona töitä tehtäessä Labranettiin piti aina ottaa VPN-yhteys, jotta tiedonsiirto onnistui.

Kehittäminen kotikoneella muuttui helpommaksi, kun kehitystekniikka vaihdettiin PHP-tekniikasta .NET-tekniikkaan. Sovellukseen tulleita kehityksiä ei enää tarvinnut siirtää FTP:n avulla ulkoiselle palvelimelle, sillä ne pystyi testaamaan lokaalisti. Tietokantayhteyden toiminta vaati kuitenkin yhä VPN-yhteyden muodostamista sovellusmuutoksia testatessa. Tämä oli silti paljon helpompi ratkaisu kuin tiedostojen

lähettely palvelimelle jokaisen muutoksen myötä. Windows-ympäristössä toimiva IIS-server oli myös paljon helpompi ympäristö pystyttää ja käyttää kuin palvelimella oleva Mod_mono -laajennus.

Ohjelmakoodin kehittämiseen käytettiin alunperin JetBrains yhtiön luomaa ohjelmaa PHP Storm, mutta .NET:iin siirtymisen myötä käyttöön otettiin Visual Studio 2013. Visual Studiossa on sisäänrakennettu IIS Express, joka helpottaa tuotetun Client-puolen koodin testausta merkittävästi. Ohjelmakoodeja ei tarvitse erikseen asentaa, jotta ne voitaisiin ajaa, sillä Visual Studio kääntää binäärit ja ajaa ne suoraan IIS Expressin päällä. Server-puolen koodit puolestaan ajetaan erikseen IIS-palvelimen päällä, josta Client-puoli niitä käyttää.

4.2 Perusidea

Toteutetun sovelluksen perusidea rakentuu pääasiassa yhden tietokantataulun hallinnan ympärille. Taulussa säilytetään kaikki työn alla olevat opinnäytetyöt ja niiden tiedot. Tauluun liittyy myös muutama alitaulu, joissa säilytetään opinnäytetöihin liittyvää monimutkaisempaa dataa, kuten tiedostoja ja muita töihin liittyviä olioita.

Sovelluksen tärkeimpiin ominaisuuksiin lukeutuu epärelevantin tiedon piilottaminen käyttäjiltä. Tämä hoidettiin antamalla kaikille käyttäjille käyttäjäroolit, jotka rajaavat, mitä tietoja millekin käyttäjäroolille näytetään.

Tämän lisäksi sovellukseen toteutettiin erillinen rajapinta ulkoisille järjestelmille. Näin mahdollistettiin sovelluksen jatkokehitys myös muille kuin web-alustoille.

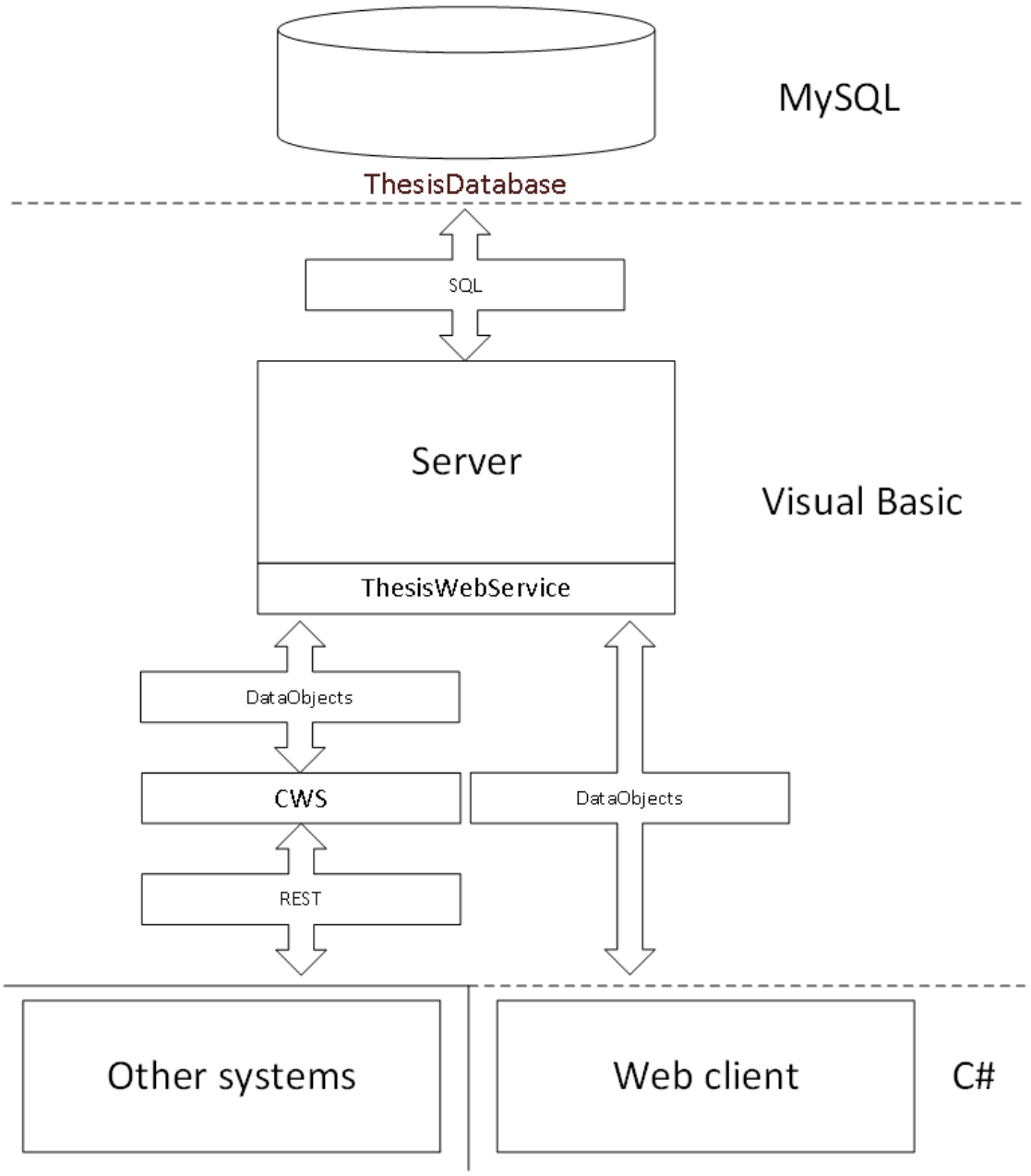
4.3 Sovelluksen rakenne

4.3.1 Yleistä

Sovelluksen toteutus aloitettiin ensin 3-tier-arkkitehtuurina, mutta se päivitettiin myöhemmin n-tier-arkkitehtuuriin. Ensimmäinen PHP-kielellä toteutettu versio nettisivusta oli suoraan yhteydessä tietokantaan. Tämä ei ollut optimaalinen ratkaisu olio-pohjaiselle modulaariselle järjestelmälle.

N-tier-arkkitehtuurin myötä Serverin käyttäminen myös muualta kuin web-palvelimelta ajettavalta käyttöliittymältä mahdollistui. Lisäksi ohjelmiston ylläpito on helpompaa, sillä teknologian vaihtamisen myötä ohjelmiston rakenne muuttui selkeämmäksi ja järjestelmällisemmäksi.

Ohjelmointikielenä Client-puolella oli käytössä C#. Server-puoli, SharedCodes ja CWS puolestaan toteutettiin Visual Basicilla (ks. Kuvio 1).



Kuvio 1. Järjestelmäkuvaus ja tekniikat

4.3.2 Client

Sovelluksen käyttöliittymä toteutettiin ASP.NET-teknologialla. Visual Studio sisältää muutamia yleisimpiä rakenteita verkkosivuille, kuten MVC5 ja Web Forms. Käytön helppouden ja yksinkertaisemman rakenteen vuoksi näistä kahdesta otettiin käyttöön Web Forms, joka sisältää sivuston perusrakenteen ja muutamia esimerkkejä kehitystyön helpottamiseksi.

MVC5 oli toinen potentiaalinen vaihtoehto kehitysmalliksi, mutta Web Forms valittiin mieluummin, sillä se oli ennalta tutumpi kuin MVC5, ja opiskeltavaa oli kuitenkin jo valmiiksi luvassa kehitystyön myötä. Muiden asioiden opiskelu priorisoitiin MVC5:n edelle, koska myös Web Forms saatiin muotoutumaan suhteellisen helposti MVC-malliin.

WebService -rajapinta

Käyttöliittymä on yhteydessä Serveriin yhden WebService -rajapinnan kautta. Serverin Webservice -rajapinnassa on metodit kaikille tärkeimmille toiminnoille, joita hallitaan kriteerien ja dataluokkien avulla. Esimerkiksi opinnäytetyöt voidaan hakea rajapinnasta seuraavalla metodilla:

```
public SharedCodes.ThesesData LoadTheses(SharedCodes.ThesesCriteria
criteria) {
    return base.Channel.LoadTheses(criteria);
}
```

ThesesCriteria luokka sisältää muuttujia, jotka määrittämällä voidaan vaikuttaa hakutuloksiin. ThesesCriteria-luokan sisältö:

```
Public Class ThesesCriteria
    Public Property ThesisId As Integer
    Public Property NameFinnish As String
    Public Property NameEnglish As String
    Public Property UserId As String
    Public Property FirstDirector As String
    Public Property SecondDirector As String
    Public Property PeerReviewer As String
    Public Property Orientation As ThesisEnums.OrientationEnum?
    Public Property StartDateStart As Date?
    Public Property StartDateEnd As Date?
```

```
    Public Property EstimatedFinishDateStart As Date?
    Public Property EstimatedFinishDateEnd As Date?
    Public Property Customer As String
    Public Property AgreementDateStart As Date?
    Public Property AgreementDateEnd As Date?
    Public Property LanguageDirectingProgress As Integer?
    Public Property EnglishSummaryProgress As Integer?
    Public Property UrkundResult As Integer?
    Public Property MaturityTest As Boolean?
    Public Property PublishedInTheseus As Boolean?
    Public Property GradeSuggestion As
ThesisEnums.GradeSuggestionEnum?
    Public Property Finished As Boolean?
    Public Property FinishDateStart As Date?
    Public Property FinishDateEnd As Date?
    Public Property Approved As Boolean?
    Public Property UserFullName As String
    Public Property FirstDirectorName As String
    Public Property SecondDirectorName As String
    Public Property PeerReviewerName As String
End Class
```

Luokat

Client-puoli sisältää omat versionsa käsiteltävistä luokista. Server-puolella olevat luokat eivät vielä toistaiseksi eroa juurikaan Client-puolen luokista, mutta ne on tarkoituksella erotettu toisistaan. Hyötynä tästä on se, että esimerkiksi Client-puolella olevan luokan laajentuessa sisäisten tarpeiden täyttämiseksi, ei tietoa tarvitse kuljettaa rajapinnan yli Server-puolelle erikseen. Sama pätee Server-puolella olevien luokkien laajenemiseen. Tieto näiden kahden puoliskon välillä kulkee aikaisemmin mainittujen SharedCodes-kirjastossa olevien dataluokkien avulla.

Lokalisointi

Kirjautunut käyttäjä voi valita millä kielellä hän haluaa käyttöliittymää käyttää. Tätä varten on tietokannan käyttäjätaulussa tieto, jonka perusteella Client-puolella näytetään lokalisoidut tekstit. Lokalisoinnin toteutus oli hieman hankalaa, eikä yhtä hyvää tapaa tuntunut löytyvän. Niin sanotun oikean toteutustavan mukaan tehtynä

olisi jokaiseen näkymään pitänyt kopioida samanlaista koodia. Täysin saman koodin kirjoittaminen useaan paikkaan on huono tapa kehittää koodia.
Onneksi lopulta lokalisoinnin toteuttamiseen keksittiin kuitenkin ratkaisu; näkymien kanssa samaan kansioon perustettiin uusi tyhjä näkymä nimeltään CultureHelper.aspx. Tähän näkymään tehtiin kaikki lokalisointiin tarvittava koodi, ja kaikki muut näkymät laitettiin periytymään tästä yhdestä näkymästä. Näin ollen kaikki näkymät saivat toimivan lokalisaation ilman tarvetta kopioidulle koodille.

Dialogit ja AJAXService

Eri näkymissä olevien taulukoiden rivejä klikkaamalla voidaan avata kyseisen rivin tiedot sisältävä dialogi. Esimerkiksi opinnäytetyön ohjaaja saa avattua ohjaamiaan opinnäytetöitä muokattavaksi tähän ikkunaan, ja tallentamaan muokatut tiedot ikkunan alalaidassa olevasta OK-napista. Tiedot olisi ollut mahdollista tallentaa suoraan taustalla napin klikkaamisesta syntyvän eventin laukaisemana, mutta jos näin oltaisiin tehty, tallennus ei olisi ollut asynkroninen. Sen sijaan dialogeista tallennettavat tiedot lähetetään AJAXService-nimiseen rajapintaan, joka vastaanottaa viestin, ja päättelee minkä tyyppinen olio ollaan tallentamassa tai päivittämässä. Oikean käsittelijän löydyttyä AJAXService antaa olion käsittelijälle, joka sitten käyttää luokan sisäistä Save-metodia tiedon tallentamiseen.
Työn liitteenä on esimerkkinä opinnäytetyön tietojen päivittämiseen liittyvät koodit.

4.3.3 SharedCodes

SharedCodes on järjestelmää varten kehitetty kirjasto, jonka binäärit on sisällytetty sekä Serverin että käyttöliittymien yhteyteen. Koska tietoa on siirrettävä Serverin ja käyttöliittymien välillä, molemmat järjestelmän osat tarvitsevat joitakin yhteneviä tietorakenteita. SharedCodes-kirjasto auttaa tässä, sillä se sisältää kaikki käytössä olevat oliot ja enumeraatiot.

Tieto kulkee käyttöliittymien ja Serverin välillä SharedCodesissa määritetyssä muodossa. Yhteisiä tietorakenteita ei kuitenkaan käytetä missään ohjelmiston osassa tiedonkäsittelyyn, ainoastaan tiedon siirtämiseen. Tämä mahdollistaa Serverillä ja käyttöliittymillä olevien olioiden eroavaisuudet. Näin esimerkiksi Serverillä voidaan

käsitellä olioissa tietoja, joita ei koskaan tarvita käyttöliittymillä, ja näin voidaan vähentää Web Servicen yli kulkevan turhan datan määrää.

4.3.4 Tietokanta

Koska järjestelmää kehitettiin aluksi LAMP-alustalle, tietokantatekniikaksi valikoitui luonnollisesti MySQL. Tekniikan vaihtuessa potentiaalisena vaikeutena oli .NET teknologian sovittaminen yhteen muun kuin Microsoftin tarjoaman tietokantatekniikan kanssa. Sopivan kirjaston löytäminen MySQL-kannan käyttöön löytyi kuitenkin suhteellisen helposti, ja tietokannan käyttämiseen tarvittavan koodin teko onnistui ilman suurempia ongelmia.

4.3.5 CWS

Client Web Service on ulkoisille järjestelmille tarkoitettu rajapinta Serverin käyttöön. Koska esimerkiksi SharedCodesin jakaminen ulkoisille järjestelmille ei onnistu, CWS tarjoaa REST-tekniikalla toimivan rajapinnan, joka ei tarvitse toimiakseen mitään jaettua osaa järjestelmän kanssa.

CWS:n käyttö onnistuu JSON-kutsuilla. Seuraavassa esimerkki uuden opinnäytetyön luomiseen tarvittavasta JSON-kutsusta:

```
{
    "UserId": "G2007",
    "NameFinnish": "Opinnäytetöiden hallintajärjestelmä",
    "NameEnglish": "Thesis manager",
    "FirstDirector": "RanAr",
    "SecondDirector": "SahKa",
    "PeerReviewer": "G2008",
    "Orientation": "ProgrammingTechnology",
    "StartDate": "2015-05-01",
    "Customer": "JAMK"
}
```

Ja CWS:n vastaanottava koodi:

```
<HttpPost()>
Public Function InsertThesis(thesisDTO As ThesisDTO) As ThesisDTO

    Dim ret As New ThesisDTO
    Dim thesis As New Thesis
    _thesisRepository = New ThesisRepository

    Try

        ThesisMapper.Map(thesis, thesisDTO)

        Dim result = _thesisRepository.Save(thesis)

        ThesisMapper.Map(ret, result)

    Catch ex As Exception

        Throw New Exception("Inserting new thesis failed", ex)

    End Try

    Return ret

End Function
```

4.4 Server

4.4.1 Yleistä

Serverin tehtävänä on yhdistää rajapinnat tietokantaan ja kartoittaa tietokannan tiedot olioihin. Lisäksi Serverillä suoritetaan kaikki olioiden raskaampi käsittely ja matemaattiset laskutoimitukset.

Serverille luotiin ainoastaan yksi rajapinta, mutta useampien rajapintojen luominen voi tulevaisuudessa olla tarpeen, mikäli ohjelmaa laajennetaan tulevien opinnäytetöiden yhteydessä. Web Servicet ovat Serverin rajapintoja käyttöliittymille. Toteutetussa sovelluksessa on kaksi käyttöliittymää; nettisivujen kautta toimiva Client ja REST-kutsuilla toimiva CWS.

Opinnäytetöiden lataaminen Serveriltä on Web Servicessä toteutettu seuraavasti:

```
Public Function LoadTheses(ByVal criteria As ThesesCriteria) As
ThesesData Implements IWebService.LoadTheses

    Dim theses As New Theses

    Try
        theses = _thesisRepository.LoadThesesByCriteria(criteria)
    Catch ex As Exception
        Throw
    End Try
```

```
    Return theses.GetStructure()

End Function
```

4.4.2 Luokat

Luokat ovat järjestelmän ydin. Kaikki tieto, mikä kulkee tietokannan ja käyttöliittymän välillä, säilytetään luokista luoduissa olioissa. Vaikka luokat ovat yksinkertaisimmillaan tietorakenteita, joihin tallennetaan ja joista ladataan dataa, voi niitä käyttää laajemminkin.

Tässä järjestelmässä oliot voivat itsenäisesti tallentaa, ladata ja päivittää vastaavia rivejä tietokannan tauluista. Olioita ei siis luoda erikseen ja tallenneta dataa niihin ulkopuolelta, vaan olemassa olevalle oliolle annetaan käsky tallentaa tai ladata tietonsa tietokannasta, ja olio itse sisäisesti tekee tämän. Oliot osaavat myös itse tehdä tallennettavan datan validoinnin, arvojen asettamisen ja SharedCodesin dataluokkiin sopivan rakenteen luomisen.

Tallentaminen kantaan on tehty hyvin yksinkertaiseksi. Esimerkiksi olio nimeltään 'thesis' voidaan luoda ja tallentaa kantaan seuraavasti:

```
Dim thesis As New Thesis() With {
    .UserId = "g2007",
    .NameFinnish = "Oppari",
    .NameEnglish = "Thesis",
    .StartDate = Date.Now
    }

thesis.Save()
```

Tämä lisää kantaan uuden opinnäytetyön annetuilla tiedoilla.

Olion päivittäminen tapahtuu samaa Save-metodia käyttämällä. Lisäksi Save-metodiin annetaan totuusarvotieto siitä, onko kyseessä olemassa olevan rivin päivitys, vai uuden rivin tallennus. Esimerkkinä tästä koodi:

```
Dim existingThesis = _thesisRepository.Load(criteria)

With existingThesis
    .NameFinnish = "Joku muu nimi"
    .NameEnglish = "Some other name"
End With

existingThesis.Save(True)
```

Mikäli tallennus kantaan ei onnistu, lentää Save-metodista ulos poikkeus, joka voidaan käsitellä myöhemmin koodissa halutulla tavalla.

Tietokannasta lataaminen tapahtuu Load-metodilla, jolle lähetetään kriteerit, joiden perusteella varsinainen tietokantakysely tehdään. Olio käy läpi annetut kriteerit ja rakentaa tietokantakyselyn niiden perusteella. Mikäli kriteeriä, kuten käyttäjän tunnusta, ei ole määritetty, jätetään se kokonaan kyselystä pois.

4.4.3 BusinessClass

Tietokannan näkymiä vastaavat luokat perivät yhteisen BusinessClass-luokan. BusinessClass-luokka sisältää tietokantayhteyden luomiseen tarvittavan IConnection-luokan esittelyn ja käyttöön tarvittavan koodin. Lisäksi BusinessClass sisältää muutamia MustOverride-metodeja, jotka toteutetaan perivissä luokissa. Näihin kuuluu tällä hetkellä Save, Validate ja GetTableName -metodit.

4.4.4 Managerit

Tärkeimmille toimille, kuten opinnäytetöiden käsittelylle tehtiin omat managerit. Managerit tarkistavat muun muassa tallennettaessa, onko tallennettava tieto jo olemassa tietokannassa, vai luodaanko kantaan uusi rivi. Lisäksi, kun olioita käsitellään managerissa, tuleva monimutkaisempi käsittely serverin puolella voidaan sisällyttää Manager-luokkiin.

4.4.5 Login-metodi

Käyttäjän kirjautuminen järjestelmään tapahtuu User-luokassa olevalla Login-metodilla. Käyttäjän tunnuksen ja salasanan perusteella tiedustellaan LabraNetin Active Directoryltä ovatko käyttäjän tunnukset oikein. Jos ovat, palautuu Active Directorysta tieto onnistuneesta kyselystä. Huomioitavaa on, että vastauksen mukana ei tule mitään muuta tietoa, kuin se, olivatko käyttäjätunnus ja salasana oikein. Tämä kuitenkin riittää uuden session luomiseen. Mikäli käyttäjä annetulla käyttäjätunnuksella löytyy jo tietokannasta, luodaan uusi sessio. Muuten kantaan luodaan uusi käyttäjä minimitiedoilla.

4.4.6 IConnection

IConnection-luokkaa käyttämällä oliot voivat tallentaa ja ladata tietoa kannasta. Tietokantakyselyiden muodostamista yhtenäistettiin ja kerättiin tämän yhden luokan

ympärille niin, että kaikki oliot toimisivat samalla tavalla. Luokassa on muutama metodi, jotka ottavat parametreinaan SQL-tekstimuuttujan, sekä listan objekteja. Esimerkiksi metodi ExecuteQuery on esitelty seuraavasti:

```
Public Function ExecuteQuery(ByVal sql As String, ByVal params As
Object) As List(Of Dictionary(Of String, Object)) _
        Implements IConnection.ExecuteQuery
```

Metodi palauttaa tietokannasta tulevan vastauksen muokattuna IConnection-luokkaa käyttävien olioiden ymmärtämään muotoon.

4.5 Tietokannan taulut

Tietokannan tauluihin tallennetaan lähes kaikki tieto, mitä järjestelmässä käsitellään. Tärkein näistä on taulu, joka sisältää opinnäytetyöt ja niihin liittyvän datan. Lisäksi käyttäjätiedot ja tiedot tallennetuista tiedostoista ovat tietokannan tauluissa.

Taulut nimettiin formaatilla: 'tbl_taulunNimi', sillä joillain tietokantaan luoduilla näkymillä ja tauluilla olisi ollut samat nimet, mikäli tauluja ei olisi erotettu näkymistä jollain etuliitteellä.

4.5.1 tbl_thesis

Tietokannan taulu tbl_thesis sisältää kaikki opinnäytetyöt. Opinnäytetyön tekijän, ensimmäisen ja toisen ohjaajan sekä työn vertaisarvioijan käyttäjätunnukset ovat linkitettyinä tbl_user-tauluun foreign key -liitoksella, jolloin opinnäytetyötaulussa ei voi olla opinnäytetöitä väärillä henkilöliitoksilla. Tämän taulun vyörytyssäännöt selittyvät helpoiten kuvalla (ks. Kuvio 2).

Näppäinnimi	Sarakkeet	Viitetaulu	Vierassarakkeet	On UPDATE	On DELETE
fk_tbl_thesis_tbl_orientation	orientation	tbl_orientation	orientation	CASCADE	RESTRICT
fk_tbl_thesis_tbl_user	userId	tbl_user	userId	RESTRICT	RESTRICT
fk_tbl_thesis_tbl_user1	firstDirector	tbl_user	userId	CASCADE	SET NULL
fk_tbl_thesis_tbl_user2	secondDir...	tbl_user	userId	CASCADE	SET NULL
fk_tbl_thesis_tbl_userPeer	peerRevie...	tbl_user	userId	CASCADE	SET NULL

Kuvio 2. tbl_thesis vyörytyssäännöt

4.5.2 tbl_user

Tbl_user on käyttäjätaulu, johon tallennetaan käyttäjien tiedot. Erikoista verrattuna muiden järjestelmien käyttäjätauluihin on se, että tässä taulussa ei pidetä salasanoja tallessa. Opinnäytetyöjärjestelmän käyttäjien salasanat tarkistetaan sisään kirjauduttaessa Labranetin Active Directory -järjestelmästä LDAPin avulla, jolloin Labranetin käyttäjän salasanan vaihtuessa myös opinnäytetyöjärjestelmään pääsee kirjautumaan päivitetyllä salasanalla.

4.5.3 tbl_roles

Käyttäjien roolit pidetään tallessa tbl_roles-taulussa. Rooleja ei voi tallentaa käyttäjätauluun, koska yhdellä käyttäjällä voi olla useampikin rooli. Kun käyttäjä haetaan kannasta, samalla haetaan roolit, joita käyttäjälle on lisätty.

4.5.4 tbl_rolePool

Tbl_rolePool-taulussa pidetään listaa rooleista, joita käyttäjille voi antaa. Tämä taulu on liitetty tbl_roles-tauluun siten, että käyttäjille ei voi antaa rooleja, joita ei ole määritelty tbl_rolePool-taulussa. Samalla tbl_rolePool antaa suomenkielisen selitteen rooleille, kun käyttäjiä ladataan tietokannasta.

4.5.5 tbl_orientation

Koulutusohjelmien listaa pidetään yllä tbl_orientation-taulussa, josta valitaan opinnäytetyölle kuuluva koulutusohjelma. Lisäämällä tähän tauluun rivejä, voidaan lisätä koulutusohjelmavaihtoehtoja.

4.5.6 tbl_peerReview

Vertaisarvioijan tekemä arviointi tallennetaan tbl_peerReview-tauluun ja linkitetään tbl_thesis-taulussa olevaan opinnäytetyöhön.

4.6 Tietokannan näkymät

Tietokannassa on tietokantataulujen lisäksi näkymiä, joilla voidaan helposti laajentaa ja supistaa tietokantahakujen tuloksia. Esimerkiksi rooleja haettaessa roolien

suomenkieliset nimet voidaan hakea automaattisesti tbl_rolePool-taulusta. Kaikki tietokantahaut Serveriltä kohdistuvat tietokantanäkymiin, eivät ikinä suoraan tietokantatauluihin. Olioita tallennettaessa näkymät tosin ohitetaan, sillä osa näkymistä on koostettu useamman taulun tiedoista liitoslauseilla.

5 Tulokset

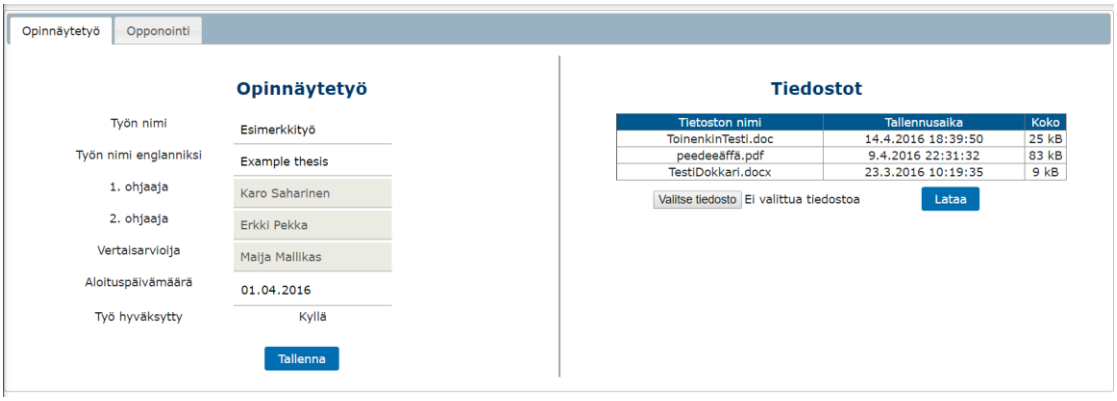
5.1 Yleistä

Tämän opinnäytetyön tuloksena saatiin järjestelmä, jonka avulla opinnäytetöitä voidaan hallinnoida tehokkaammin ja helpommin kuin aiemmin. Opiskelijat pääsevät itse näkemään oman opinnäytetyönsä tilan ja tallentamaan muiden opiskelijoiden opinnäytetöistä tekemänsä opponointiraportit. Lisäksi opinnäytetöiden ohjaajat näkevät ja pystyvät muokkaamaan ohjaamiensa opiskelijoiden opinnäytetöitä. Koulutuspäälliköt voivat seurata opinnäytetöiden tekoon liittyvää statistiikkaa ja muita tarvitsemiaan tietoja.

Alkuperäisestä vaatimusmäärittelystä saatiin toteutettua lähes kaikki suunniteltu toiminnallisuus. Vähimmälle kehitystyölle jäi koulutuspäällikön näkymä. Alustavissa vaatimusmäärittelyissä kuvattua koulutussihteerin näkymää ei toteutettu ollenkaan ajan puutteen vuoksi.

5.2 Opiskelijan näkymä

Opiskelijan näkymään (ks. Kuvio 3) saatiin toteutettua kaikki alunperin suunnitellut ominaisuudet. Opinnäytetöiden tietojen muokkausosio jäi vielä hyvin keskeneräiseksi. Tiedostojen palvelimelle lataus, palvelimelta lataus ja poistaminen saatiin kuitenkin toteutettua kokonaisuudessaan.



Kuvio 3. Opiskelijan näkymä - Opinnäytetyö-välilehti

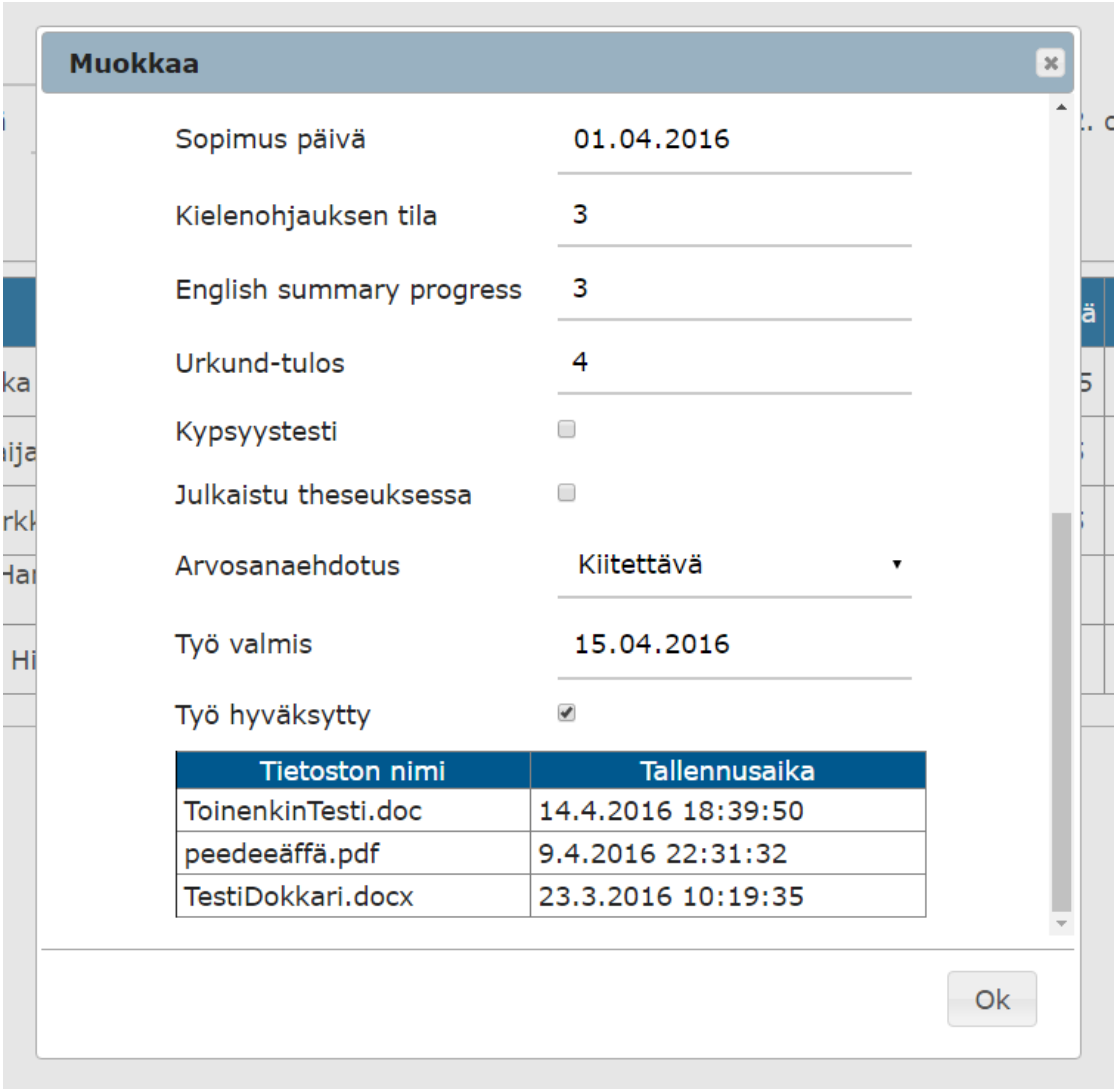
Toisella välilehdellä oleva opponointidialogi tuli valmiiksi kehitystyön loppuvaiheessa.

5.3 Ohjaajan näkymä

Ohjaajan näkymä (ks. Kuvio 4) tuli suurelta osin valmiiksi, lukuun ottamatta hakukriteerien visuaalista ilmettä. Hakukriteerien käyttäminen toimii kuten pitääkin, mutta visuaalinen ilme voisi olla hieman parempi, ja kentät selkeämmin järjestelty.

Kuvio 4. Ohjaajan näkymä

Ohjaajan klikatessa listalla olevaa riviä, aukeaa kyseisellä rivillä oleva opinnäytetyö muokkausnäkymään. Näkymässä näkyy myös työhön liittyvät opiskelijan lataamat tiedostot (ks. Kuvio 5).



Kuvio 5. Opinnäytetyön ohjaajan näkymä - Dialogi

5.4 Koulutusvastaavan näkymä

Koulutusvastaavan näkymän toteutuksesta ei tullut yhtä perusteellista, kuin opiskelijan ja opinnäytetyön ohjaajan näkymistä. Koulutusvastaava voi nyt hyväksyä opinnäytetöiden aihe-ehdotuksia ja määrittää ohjaajia opinnäytetöille. Lisäksi koulutusvastaava saa näkyviin kuvaajan siitä, millainen trendi opinnäytetöiden aloittamisessa ja valmistumisessa on havaittavissa.

Toiminnallisuus, jossa koulutussihteeri voisi määrittää ohjausmääriä opinnäytetöiden ohjaajille, joita koulutusvastaava sitten kuluttaisi määrätessään ohjaajia töille, jäi kokonaan toteuttamatta järjestelmästä. Tämä toiminnallisuus ei kuitenkaan ole vitaali järjestelmän kannalta, joten sen kehitys voidaan jättää jatkokehittäjien huoleksi.

5.5 Koulutuspäällikön näkymä

Koulutuspäällikön näkymä jäi eniten kesken kaikista näkymistä, lukuun ottamatta näkymiä joita ei toteutettu ollenkaan, kuten opintosihteerin näkymä. Näkymään tehtiin vain kuvaajan piirto. Kaikista järjestelmään syötetyistä opinnäytetöistä haetaan kuusi kuukautta taaksepäin valmistuneet työt sekä kuusi kuukautta eteenpäin valmistuvaksi arvioidut työt. Haettujen tietojen perusteella piirretään kuvaaja valmistuneista ja valmistuvista opinnäytetöistä.

5.6 Admin-näkymä

Admin-näkymä tuli valmiiksi lähes vaatimusten mukaisesti. Ainoastaan opinnäytetöiden poistamista ei toteutettu, joten opinnäytetöiden poistaminen järjestelmästä ei ole toistaiseksi mahdollista, lukuun ottamatta suoraa tietokannan muokkaamista. Muuten admin-roolilla oleva käyttäjä pystyy hallitsemaan lähes kaikkea tietokantaan tallennettavaa dataa.

6 Pohdinta

6.1 Yleistä

Opinnäytetyön päätavoitteena oli siirtää opinnäytetöiden hallinta vanhentuneesta mallista nykyaikaisempaan web-pohjaiseen ympäristöön. Opinnäytetyön myötä ASP.NET-teknologian perusteet palautuivat mieleen, ja paljon uuttakin tuli opittua. Lisäksi kokonaisen järjestelmän rakentaminen tyhjästä oli hyvää opetusta kehitettävän rakenteen eri komponenttien toiminnasta ja liitoksista toisiinsa.

6.2 Prosessin kritiikki

Turhin kehitykseen liittyvä ylimääräinen työ oli ehdottomasti teknologian vaihdosta aiheutunut koodaustyön hukkaan meneminen. Itse teknologian vaihto ei ollut turha liike, mutta ensimmäinen versio olisi pitänyt hylätä paljon aikaisemmin. Tämä varmasti vaikutti järjestelmän valmiuden asteeseen. Mikäli järjestelmän tekniikka

olisi vaihdettu aikaisemmassa vaiheessa, olisi kehitystyötä varmasti voitu saada paljon pidemmälle.

Omasta mielestäni palvelimen vaihtaminen LAMP-serveristä Windows Serveriksi olisi pitänyt tehdä. Windows Serverin pystyttäminen ja vaadittujen ohjelmistojen asentaminen ja toimintaan saattaminen eivät ole kovinkaan vaativia tehtäviä. Asensin omalle koneelleni IIS-palvelimen ja muut tarvittavat osat noin tunnin aikana, eikä Labranetissä pyörivän palvelimen asentaminen varmastikaan olisi tuottanut yhtään enempää vaivaa. En tosin täysin kattavasti tiedä mitä kaikkea palvelinpuolen kehittäjä oli jo LAMP-serverillä tehnyt, että olisiko palvelinohjelmiston vaihtaminen tuottanut muuten kohtuuttomasti lisätyötä.

6.3 Sovelluksen kritiikki

Suurimpana harmin aiheena sovelluksessa on epäilemättä se, että monia edistyneempiä ominaisuuksia jäi toteuttamatta ajan rajallisuuden ja oman opinnäytetyöni osuuden päättymisen takia. Tietenkin tämä antaa mahdollisuuksia uusien opinnäytetöiden luomiselle. Silti asioiden loppuun saattaminen olisi ollut mielestäni paljon toivotumpi vaihtoehto, kuin järjestelmän keskeneräiseksi jättäminen.

Sovelluksen toiminta voisi myös olla nopeampaa. Tietorakenteiden mappaus Serverin ja Clientin välillä kulkeviin struktuureihin ja takaisin vie väkisinkin hieman aikaa, jos tietoa käsitellään suurempia määriä. Tätä voisi parantaa NHibernate-teknologian käyttöönotolla, jota käyttämällä olisi helpompi hallita tietoja, joita tietokannasta ladataan niin sanotulla Lazy load -tekniikalla. NHibernate itse lataa tietoja kannasta sitä mukaan kun niitä tarvitaan, näin ollen minimoiden järjestelmässä liikkuvan turhan datan määrän.

Lisäksi tietoturvan taso ohjelmassa ei ole paljon perusteita korkeampi. Toistaiseksi taso on varmasti riittävä, sillä SQL-injektioilta on suojauduttu ja ohjelmaa ei saa rikottua vahingossa. Ohjelman tietoturvan laajentamisesta voisi tehdä melkein yhden kokonaisen opinnäytetyön. Ainakin Client-puoli, nettipalvelin ja CWS voisivat kaikki hyötyä tietoturvan tason parantamisesta.

6.4 Tulevaisuus

Järjestelmää voi kehittää vielä monella tapaa muiden opinnäytetöiden osana. Listasin loppuun vielä muutamia ideoita, miten itse lähtisin kehittämään olemassa olevaa ohjelmaa eteenpäin. Toivomuksena tietenkin olisi, että ohjelma ei kuolisi käyttökelvottomana tai liian suppeana, vaan sitä kehitettäisiin kohti parempaa kokonaisuutta.

6.4.1 Client/Server kehitys

Järjestelmän potentiaalista jäi osia toteuttamatta, kuten opinnäytetöiden ohjaajien ohjausallokaatioiden syöttö järjestelmään, ja niiden käyttäminen. Lisäksi kielen ohjaajan näkymä oli jossain suunnitelmassa mukana ennen varsinaisia määrittelyjä, mutta se jätettiin pois vaatimusmäärittelyistä heti alkuvaiheessa, jottei toteutettavasta järjestelmästä tulisi liian laaja. Näiden ominaisuuksien toteuttaminen voisi olla ensimmäisiä asioita joita kannattaa tehdä, jotta itse perusohjelma olisi parhaassa toimintakunnossaan.

6.4.2 CWS

Client Web Servicen loppuun kehittäminen jonkin muun opinnäytetyön ohessa on myös hyvä opinnäytetyön aihe. Tähän sopiva työ voisi olla esimerkiksi Android/iOS/Windows Phone -applikaatio, joka on yhteydessä Serveriin CWS:n kautta. Tätä varten pitäisi toteuttaa CWS:ään tarvittavat metodit Serverin käyttämiseen. Tämä on juuri se tarkoitus, jota silmällä pitäen CWS lisättiin ohjelmiston kokoonpanoon. Sen ei ole tarkoituskaan olla vielä toimiva kokonaisuus, vaan enemmänkin oljenkorsi opastamaan kännykkä-applikaation tekijää rajapinnan toteuttamisessa esimerkin avulla.

6.4.3 Tietoturva

Tietoturvan kehitys kaikkien olemassa olevien palasten osalta on hyvä ehdotus tulevaksi kehitysprojektiksi. Client-puolen tietoturvan tason nosto lienee tärkein osa kehitystä. Toisena CWS:n tietoturvan varmistaminen, ja kolmantena palvelimen

tietoturva. Yksityiskohdat jäävät tietenkin työn tekijän mietittäväksi, mutta näistä voisi ainakin aloittaa.

6.4.4 Crystal Reports

Crystal Reports on SAP-ohjelmistoyrityksen luoma raportointityökalu, jonka avulla voidaan luoda erilaisia raportteja tietokannassa säilytettävästä datasta. Ymmärtääkseni opinnäytetöiden tiloista, valmistumisennusteista ja muista tämän kaltaisista tiedoista voisi olla hyvä luoda raportteja esimerkiksi kouluhallitukselle tai muille ylemmille tahoille. Vaikka Crystal Reports ei ehkä omasta mielestäni ole paras raportointityökaluna, voi sen käytön opiskelusta olla hyötyä tulevaa työelämää varten.

Käytännössä kyseessä olevalla ohjelmalla luodaan raporttitiedosto, johon määritellään, mitä dataa raportille halutaan ja missä muodossa. Sitten käyttöliittymältä annetaan kriteerit, kuten aikavälit, ja tulostetaan raportti. Sen jälkeen Crystal Reports luo järjestelmästä tai tietokannasta saamansa datan pohjalta esimerkiksi PDF-tiedoston, jossa halutut tiedot on esitetty halutussa muodossa.

Lähteet

Mod_mono compabilities. 2016. Viitattu 17.4.2016.
http://www.mono-project.com/docs/about-mono/compatibility/

Jyväskylän ammattikorkeakoulun verkkosivut: JAMKin hallinto. 2016.
Viitattu 26.4.2016
http://www.jamk.fi/fi/Tietoa-JAMKista/JAMKin-hallinto/

Jyväskylän ammattikorkeakoulun verkkosivut: Teknologiayksikkö. 2016.
Viitattu 26.4.2016
http://www.jamk.fi/fi/Tietoa-JAMKista/Teknologiayksikko/

Jyväskylän ammattikorkeakoulun verkkosivut: Tutustu JAMKiin. 2016.
Viitattu 26.4.2016
http://www.jamk.fi/fi/Tietoa-JAMKista/Tutustu-JAMKiin/

Liitteet

Liite 1. AJAXService.svc

```
[ServiceContract]
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class AjaxService : WebService
{
    [WebInvoke(Method = "POST", BodyStyle =
WebMessageBodyStyle.WrappedRequest)]
    [WebMethod(EnableSession = true)]
    [OperationContract]
    public void HandleAjax(string sender, string data, string obj, string
crit)
    {
        var resolvedData = ResolveData(data);

        IAjaxHandler handler = GetHandler(sender);
        handler.handleAjax(resolvedData);

        UpdateSessionState(sender, obj, crit);
    }

    private void UpdateSessionState(string sender, string obj, string crit)
    {
        if (String.IsNullOrEmpty(obj) || String.IsNullOrEmpty(crit)) return;
        switch (sender)
        {
            case "thesis":
                ThesesModel theses = (ThesesModel) Session[obj];
                ThesesCriteria tCriteria = (ThesesCriteria) Session[crit];
                theses.Load(tCriteria);
                Session[obj] = theses;
                break;
            case "user":
                UsersModel users = (UsersModel)Session[obj];
                UsersCriteria uCriteria = (UsersCriteria)Session[crit];
                users.Load(uCriteria);
                Session[obj] = users;
                break;
            default:
                //No handling
                break;
        }
    }

    private Dictionary<String, Object> ResolveData(string data)
    {
        if (String.IsNullOrEmpty(data))
        {
            throw new Exception("Given data object was empty.");
        }

        var dataEntries = data.Split('/').ToList();

        // Poista ylimääräiset entryt
        var entriesToRemove = dataEntries.Where(e => e.Split(':')[0] ==
"on").ToList();
        foreach (var e in entriesToRemove) { dataEntries.Remove(e); }

        var ret = new Dictionary<String, Object>();

        foreach (var e in dataEntries.Select(entry => entry.Split(':')))
        {
            if (String.IsNullOrEmpty(e[0]))
            {
                throw new Exception("Values missing!");
```

```
            }
            if (String.IsNullOrEmpty(e[1]))
            {
                e[1] = null;
            }
            ret.Add(e[0],e[1]);
        }
        return ret;
    }

    private static IAjaxHandler GetHandler(string sender)
    {
        switch (sender)
        {
            case "thesis":
                return new ThesisHandler();
            case "user":
                return new UserHandler();
            case "peerReview":
                return new PeerReviewHandler();
            default:
                throw new Exception("No handler found.");
        }
    }
}
```

Liite 2. ThesisHandler.cs

```
public class ThesisHandler : IAjaxHandler
{
    public void handleAjax(Dictionary<string, object> data)
    {
        object outVal;

        ThesisModel thesis = new ThesisModel();
        thesis.NameFinnish = (String) data["ThesisNameFinnish"];
        thesis.NameEnglish = (String) data["ThesisNameEnglish"];
        thesis.UserId = (String) data["ThesisUserId"];
        thesis.Orientation = (ThesisEnums.OrientationEnum)
Int32.Parse((String) data["ThesisOrientation"]);
        thesis.StartDate = DateTime.Parse((String) data["ThesisStartDate"]);
        thesis.Customer = (String) data["ThesisCustomer"];
        thesis.GradeSuggestion =
(ThesisEnums.GradeSuggestionEnum)Int32.Parse((String)data["ThesisGradeSuggest
ion"]);
        if (data.TryGetValue("ThesisId", out outVal) && outVal != null) {
thesis.ThesisId = int.Parse((string)outVal); }
        if (data.TryGetValue("ThesisFirstDirector", out outVal) && outVal !=
null) { thesis.FirstDirector = (string)outVal; }
        if (data.TryGetValue("ThesisSecondDirector", out outVal) && outVal !=
null) { thesis.SecondDirector = (string)outVal; }
        if (data.TryGetValue("ThesisPeerReviewer", out outVal) && outVal !=
null) { thesis.PeerReviewer = (string)outVal; }
        if (data.TryGetValue("ThesisEstimatedFinishDate", out outVal) &&
outVal != null) { thesis.EstimatedFinishDate =
DateTime.Parse((String)outVal); }
        if (data.TryGetValue("ThesisAgreementDate", out outVal) && outVal !=
null) { thesis.AgreementDate = DateTime.Parse((String)outVal); }
        if (data.TryGetValue("ThesisLanguageDirectingProgress", out outVal)
&& outVal != null) { thesis.LanguageDirectingProgress =
int.Parse((string)outVal); }
        if (data.TryGetValue("ThesisEnglishSummaryProgress", out outVal) &&
outVal != null) { thesis.EnglishSummaryProgress = int.Parse((string)outVal);
}
        if (data.TryGetValue("ThesisUrkundResult", out outVal) && outVal !=
null) { thesis.UrkundResult = int.Parse((string)outVal); }
        if (data.TryGetValue("ThesisMaturityTest", out outVal) && outVal !=
null) { thesis.MaturityTest = ResolveBooleanFromString(outVal); }
        if (data.TryGetValue("ThesisPublishedInTheseus", out outVal) &&
outVal != null) { thesis.PublishedInTheseus =
ResolveBooleanFromString(outVal); }
        if (data.TryGetValue("ThesisFinished", out outVal) && outVal != null)
{ thesis.Finished = DateTime.Parse((String)outVal); }
        if (data.TryGetValue("ThesisApproved", out outVal) && outVal != null)
{ thesis.Approved = ResolveBooleanFromString(outVal); }

        try
        {
            thesis.Save();
        }
        catch (SoapException ex)
        {
            throw new Exception("Tietojen tallennus epäonnistui", ex);
        }
    }

    bool ResolveBooleanFromString(object b)
    {
        return (string)b == "true";
    }
}
```